

COMPUTING THE BOUNDED SUBCOMPLEX OF AN UNBOUNDED POLYHEDRON

SVEN HERRMANN, MICHAEL JOSWIG, AND MARC E. PFETSCH

ABSTRACT. We study efficient combinatorial algorithms to produce the Hasse diagram of the poset of bounded faces of an unbounded polyhedron, given vertex-facet incidences. We also discuss the special case of simple polyhedra and present computational results.

1. INTRODUCTION

The *bounded subcomplex* of an (unbounded) convex polyhedron, is its set of bounded (or, equivalently, compact) faces, partially ordered by inclusion. Polytopal complexes of this type arise in several situations each of which is interesting in its own right. Prominent examples are the *tight spans* of finite metric spaces of Dress [11], see also Isbell [16], and the *tropical polytopes* of Develin and Sturmfels [9]. The purpose of this note is to present algorithms to deal with such objects.

Typically, in applications a bounded complex of a polyhedron P is given only implicitly by a set of inequalities defining P . The primary goal is to establish algorithms to make the bounded subcomplex explicit from this input. A direct approach is to enumerate the full face lattice $\mathcal{L}(\overline{P})$ of a polytope \overline{P} projectively equivalent to P (which exists if P is pointed) and to filter it to obtain $\mathcal{B}(P)$, the poset of bounded faces. However, since $\mathcal{L}(\overline{P})$ often is much larger than $\mathcal{B}(P)$, this is not efficient.

One natural approach is to start with a (dual) convex hull computation which also yields the vertex-facet incidences. Here we focus on combinatorial algorithms which take these vertex-facet incidences as input. Other approaches are discussed briefly. Our Algorithm 1 in Section 3.1 is a modification of a combinatorial algorithm for face lattice enumeration [20]. It uses the vertex-facet incidences of \overline{P} . This method of *selective generation* is best possible, in the sense that its running time is linear in the size of $\mathcal{B}(P)$. If only the vertex-facet incidences of P are given, that is, no information about the unbounded edges is present, one can still compute $\mathcal{B}(P)$, although at the cost of a higher running time (quadratic in the size of $\mathcal{B}(P)$). The corresponding Algorithm 2 is based on interleaving the computation of the bounded faces with an incremental computation of the poset's Möbius function.

For the sake of simplicity of exposition we usually consider complexity questions in the RAM model. However, it is easy to modify each of our results such that the bit complexity can be determined: The only potential source of non-polynomiality in terms of RAM complexity arises from calling LP type oracles. Therefore, whenever necessary, we explicitly mention the relevant sizes of the linear programming problems which need to be solved.

If additional structural information is available, specialized algorithms come into focus. For instance, in the case where the polyhedron P is simple, it turns out that $\mathcal{B}(P)$ is determined by the (directed) vertex-edge-graph of P , see [15, Section 2]. This is an unbounded version of a result of Blind and Mani [5] obtained by applying techniques due to Kalai [21]. In Section 4, we employ the reverse-search approach of Avis and Fukuda [3] to generate the (directed) vertex-edge graph of $\mathcal{B}(P)$ from an inequality description of P , provided that P is simple. This can be used to either construct the bounded faces or to efficiently compute the face numbers of P and $\mathcal{B}(P)$.

In Section 5, computational experiments with Algorithm 1 are presented. We investigate five different cases in which it is interesting to compute the bounded subcomplex. The computations show that some (surprisingly) large instances can be handled. The paper closes with a list of open problems.

We are grateful to an anonymous referee for meticulous reading and for requiring a correction in Algorithm 2. This also contributed to a cleaner description of this method.

2. PRELIMINARIES AND NOTATION

Let P be a polyhedron. The lattice of all faces (*face lattice*) of P is denoted by $\mathcal{L}(P)$. We define $\varphi(P) = |\mathcal{L}(P)|$, that is, the number of all faces of P . Moreover, let n be the number of vertices of P and m be its number of facets.

A polyhedron is called *pointed* if it does not contain an entire affine line. In the sequel we always assume that P is pointed but unbounded. For our purposes this does not mean any loss of generality, since the bounded subcomplex of a polyhedron is non-empty if and only if it is pointed. In this case, the polyhedron P is projectively equivalent to a polytope \overline{P} . Each such polytope is called a *projective closure* of P . A projective closure of P has a special face F_∞ (the *far face*) corresponding to the face at infinity. Fixing an admissible projective transformation γ which maps P to \overline{P} , the face F_∞ is the unique maximal face among the faces of \overline{P} that are not images of faces of P under γ . Note that the combinatorial type of F_∞ depends on the geometry of P , not only its combinatorial structure, and that its dimension can be any number in $\{0, \dots, \dim P - 1\}$.

We begin with the description of an algorithm to compute the polytope \overline{P} from an inequality description of $P \subseteq \mathbb{R}^d$. We refer to Ziegler's monograph [26] for a general discussion of admissible projective transformations and [19, §3.4] for an explicit construction. First, we compute one vertex v of P by solving a linear program similar to Phase I of the Simplex Algorithm. Such a vertex exists as P is pointed. In a second step we determine the active constraints at v , that is, those inequalities which are satisfied with equality at v . Let τ be the affine transformation moving v into the origin. In the image $\tau(P)$ the constraints active at 0 correspond to homogeneous linear equations. Any subset of those constraints defines a polyhedral cone containing the translated polyhedron $\tau(P)$. Among the constraints active at 0 we choose a dual basis by Gauß–Jordan-elimination. Next we pick a linear transformation ρ mapping this dual basis to the basis which is dual to the standard basis of \mathbb{R}^d . Then the image $\rho(\tau(P))$ is a polyhedron with

vertex 0 such that the boundary hyperplanes of the positive orthant define valid inequalities. In particular, $\rho(\tau(P))$ is contained in the positive orthant.

In general, intersecting the image of a polyhedron in \mathbb{R}^d under a projective transformation in $\text{PGL}_d(\mathbb{R})$ with \mathbb{R}^d may yield something non-convex. However, if $M \in \text{GL}_{d+1}(\mathbb{R})$ is an invertible matrix with non-negative coefficients, then the induced projective linear transformation $[M] \in \text{PGL}_d(\mathbb{R})$ maps the positive orthant into itself, and the image of any polyhedron inside is again a polyhedron. Now let μ be the unique projective linear transformation which fixes each coordinate hyperplane of \mathbb{R}^d and which additionally maps the hyperplane at infinity to the affine hyperplane $\sum x_i = 1$. This transformation can be written as a $(d+1) \times (d+1)$ -matrix of zeroes and ones. The polyhedron

$$\overline{P} := \mu(\rho(\tau(P)))$$

is contained in the simplex given by $\sum x_i \leq 1$ and $x_i \geq 0$ for $i \in [d]$. In particular, \overline{P} is bounded and projectively equivalent to P . The equation $\sum x_i = 1$ defines a hyperplane supporting \overline{P} , and its intersection with \overline{P} is the far face F_∞ . For given matrix A and right hand side b , let $\ell(A, b)$ be the time complexity of solving the linear feasibility problem to find a point x satisfying $Ax \leq b$. The discussion above can be summarized as follows.

Proposition 1. *If $P = P(A, b)$ is given by inequalities, there exists an explicit algorithm to compute \overline{P} in $O(\ell(A, b))$ time.*

The unbounded edges of P correspond to edges of \overline{P} that contain exactly one vertex in F_∞ . The vertices of P correspond to vertices of \overline{P} that are not contained in F_∞ . Via these relations it is easy to describe the face lattice of P in terms of \overline{P} and vice versa. The following relationship holds between the size $\varphi = \varphi(P)$ of the face posets of P and the size $\overline{\varphi} = \varphi(\overline{P})$ of its projective closure.

Lemma 2. *We have $\overline{\varphi} \leq 2(\varphi - 1)$.*

Proof. For each unbounded face F of P let $s(F)$ be the intersection of F with the hyperplane at infinity. Then s is a map from the set of unbounded faces of P to the set of non-empty faces of the far face F_∞ which is surjective. This proves that \overline{P} has at most twice as many non-empty faces as P . To establish our slightly stronger claim, observe that P must have at least one vertex, since we assumed P to be pointed. \square

The above bound is tight: For \overline{P} consider any pyramid with its basis as its special face. In view of the obvious inequality $\varphi \leq \overline{\varphi}$ the lemma shows that $\overline{\varphi} \in \Theta(\varphi)$; hence in statements about asymptotic complexity, φ and $\overline{\varphi}$ can be used interchangeably.

We label the facets of \overline{P} from 1 to \overline{m} and identify each facet with its index in $\mathcal{F} := \{1, \dots, \overline{m}\}$. Similarly, label the vertices of \overline{P} from 1 to \overline{n} , and identify each vertex with its index in $\mathcal{V} := \{1, \dots, \overline{n}\}$. Let $I \in \{0, 1\}^{\overline{m} \times \overline{n}}$ be a *vertex-facet incidence matrix* of \overline{P} with entries $I(f, v)$, that is, $I(f, v) = 1$ if facet f contains vertex v , and $I(f, v) = 0$ otherwise. Denote by α the number of incidences, that is, the number of ones in I . The face lattice of a polytope is atomic and co-atomic: Every face of \overline{P} is uniquely defined by its set of vertices and the set of facets it is contained in, respectively. In the

following, we will usually identify a face with its set of vertices and store each such set by its characteristic vector (bitset). Hence, the storage requirement per face is in $O(\bar{n})$. Modifications for other formats are straightforward – see also Section 5 below.

The computation of an incidence matrix for a polytope given by inequalities requires to generate the set of vertices, which can be exponentially many in the number of inequalities. There are several algorithms to generate the vertices of a polytope, see Seidel [23] for an overview and Avis et al. [2] as well as [17] for discussions of the complexity of vertex generation algorithms. In fact, it is currently unknown whether the vertices of a *polytope* can be generated in polynomial time in the combined size of the input and output. It is, however, known that it is NP-complete to decide whether the list of vertices of a *polyhedron* is complete (unbounded edges are ignored), see Khachiyan et al. [22] and Boros et al. [7]. For *simple* d -polytopes, in which each vertex is contained in exactly d facets, the *reverse search* algorithm by Avis and Fukuda [3] produces the vertices in time $O(d \cdot \bar{m} \cdot \bar{n})$.

Let \mathcal{D} be the *Hasse diagram* of \bar{P} . That is, \mathcal{D} is a directed rooted acyclic graph whose nodes correspond to the elements of $\mathcal{L}(\bar{P})$. If N_H, N_G are nodes in \mathcal{D} and H and G are the corresponding faces of \bar{P} , then there is an arc (N_H, N_G) in \mathcal{D} if and only if $H \subset G$ and $\dim(G) = \dim(H) + 1$. Note that we distinguish between a face F and its corresponding node N_F in \mathcal{D} , because this makes a difference in Algorithms 1 and 2 below.

3. THE FACE POSET OF THE BOUNDED SUBCOMPLEX

The partially ordered set (poset) of the bounded faces of an unbounded polyhedron P is denoted as $\mathcal{B}(P)$, and we call it the *bounded subcomplex* of P . As a polytopal complex, $\mathcal{B}(P)$ is not pure in general: it may have maximal faces of various dimensions. The bounded subcomplex is non-empty if and only if P is pointed. We denote the size of $\mathcal{B}(P)$ by φ' in the following.

We primarily address the problem to compute $\mathcal{B}(P)$, where P or \bar{P} is given in terms of a vertex-facet incidence matrix. Other input is discussed at the end of this section. The output $\mathcal{B}(P)$ should be given by its Hasse diagram, which is just a subgraph of the Hasse diagram of \bar{P} . Additionally or alternatively, the nodes, that is, faces, might be labeled by either the set of facets the corresponding face is contained in or by its set of vertices (this representation is unique, since the faces are bounded); this requires an additional overhead of at least $O(n)$ or $O(m)$ per face, respectively.

For the case where $\varphi' \approx \varphi$ and a vertex-facet incidence matrix $I \in \{0, 1\}^{\bar{m} \times \bar{n}}$ of \bar{P} is available, the direct approach is to apply the algorithm of [20] to generate the Hasse diagram of \bar{P} in $O(\bar{n} \cdot \alpha \cdot \varphi)$ time if the faces are represented by their vertices; this algorithm will be modified in the next section. We then remove the unbounded faces and their incident arcs by checking whether the intersection of each face with F_∞ is empty or not; if the faces are stored as bitsets, this requires $O(\bar{n})$ time per face; the total running time for this step is then dominated by the generation of the Hasse diagram of \bar{P} .

Alternatively, one can also generate the Hasse diagram by working on the dual, using $O(\bar{m} \cdot \alpha \cdot \varphi)$ time; in this case, faces are represented by the facets

Input: incidence matrix I of \overline{P} , far face $F_\infty \subset \mathcal{V}$
Output: Hasse diagram \mathcal{D} of $\mathcal{B}(P)$

- 1 initialize \mathcal{D} with N_\emptyset corresponding to the empty face
- 2 initialize the list $\mathcal{Q} \subseteq V(\mathcal{D}) \times 2^\mathcal{V}$ by (N_\emptyset, \emptyset)
- 3 **while** $\mathcal{Q} \neq \emptyset$ **do**
- 4 choose some $(N_H, H) \in \mathcal{Q}$ and remove it from \mathcal{Q}
- 5 compute the set \mathcal{G} of incident faces G with $\dim G = \dim H + 1$
- 6 **foreach** $G \in \mathcal{G}$ **do**
- 7 **if** G is bounded **then**
- 8 locate/create the node N_G corresponding to G in \mathcal{D}
- 9 **if** N_G was newly created **then**
- 10 add (N_G, G) to \mathcal{Q}
- 11 add the arc (N_H, N_G) to \mathcal{D}

Algorithm 1: The method of selective generation

they are contained in. In this case, the removal step takes $O(\alpha)$ time per face, since we have to intersect the vertex sets of F_∞ and of the facets that contain the face (storing the vertex sets in sorted lists). If the intersection is empty, the face is bounded. Thus, its running time is again dominated by the generation of the Hasse diagram.

If $\varphi' \ll \varphi$ there is a more efficient algorithm available, which we describe in the next section.

3.1. Selective Generation. In the following, we will describe an algorithm that computes (the Hasse diagram of) $\mathcal{B}(P)$ efficiently if an incidence matrix of \overline{P} is available. Our Algorithm 1 is a simple modification of the algorithm in [20], which computes the face lattice of a polytope. We provide some details for completeness and to enable a running time analysis.

The algorithm performs a graph search through the Hasse diagram of $\mathcal{B}(P)$ starting from the bottom (the empty face). We denote by $V(\mathcal{D})$ the set of nodes of \mathcal{D} . In each step we consider a face H that has not been considered before, see Step 4. Then we generate the set \mathcal{G} of all faces G with $G \supset H$, $\dim G = \dim H + 1$, that is, there exists an arc (N_H, N_G) between the nodes N_H and N_G corresponding to H and G , respectively. It is shown in [20] that the generation of \mathcal{G} can be performed in $O(\overline{n}^2) \subseteq O(\overline{n} \cdot \alpha)$ time. For each $G \in \mathcal{G}$, we can now test whether G is unbounded by checking whether its intersection with F_∞ is empty or not (in $O(\overline{n})$ time), see Step 7. If it is bounded, we proceed to generate the arc (N_H, N_G) . For this, one has to find or create the node N_G corresponding to G (Step 8). Here, we use a data structure, called *face tree* in [20], to store (bounded) faces and their corresponding nodes.

Each edge of a face tree stores a vertex. The vertices along a path from the root form a generating set for each face. Each sub-path corresponds to a sub-face. Because each face of a bounded face is bounded as well, each node of the tree corresponds to a different bounded face. The key point is that the computation of the face from the generating set takes $O(\alpha)$ time, see [20] for more details. For future reference we state the following.

Lemma 3 ([20]). *The face tree data structure allows to store bounded faces such that finding a face (or determining whether the face is not present) takes $O(\alpha)$ time with a storage requirement of $O(\bar{n} \cdot k)$, if k faces are stored.*

Returning to Algorithm 1, the while-loop in Step 3 is executed for each of the φ' bounded faces, because of the boundedness test in Step 7. Hence, Steps 4 and 5 contribute $O(\bar{n} \cdot \alpha \cdot \varphi')$ time in total. The for-loop in Step 6 is executed for each out-arc of a face in $\mathcal{B}(P)$ with respect to the Hasse diagram of \bar{P} . Since a face can have at most \bar{n} out-arcs, the for-loop is executed at most $\bar{n} \cdot \varphi'$ times. Step 7 takes $O(\bar{n}) \subseteq O(\alpha)$ time and Step 8 uses $O(\alpha)$ time, because of Lemma 3; note that since P is pointed, each facet contains at least one vertex, i.e., $\bar{n} \leq \alpha$. Thus, the for-loop contributes $O(\bar{n} \cdot \alpha \cdot \varphi')$ time. This shows the following.

Theorem 4. *Given the vertex-facet incidences of \bar{P} , the Hasse diagram of $\mathcal{B}(P)$ can be computed in time $O(\bar{n} \cdot \alpha \cdot \varphi')$.*

Remark 5. By Lemma 3, the face tree uses $O(\bar{n} \cdot \varphi')$ space. Storing faces as bitsets, the list \mathcal{Q} needs at most $O(\bar{n} \cdot \varphi')$ space. The output amounts to an additional space requirement of $O(\bar{n} \cdot \varphi')$. Thus, we need a total amount of storage of $O(\bar{n} \cdot \varphi')$.

Remark 6. Algorithm 1 is faster than the straight-forward algorithm described above if $\bar{n} \cdot \varphi' < \min\{\bar{n}, \bar{m}\} \cdot \varphi$.

In Section 4 below, we discuss the special case of simple polyhedra.

3.2. Selective Generation Without Knowing the Face at Infinity.

If only a vertex-facet incidence matrix $I \in \{0, 1\}^{m \times n}$ of an unbounded (pointed) polyhedron P is known, but no information about the unbounded edges is available, one can still produce (the Hasse diagram of) $\mathcal{B}(P)$ as follows.

Let $\mathcal{P}(P) := \{\text{vert}(F) : F \text{ proper face of } P\} \cup \{\emptyset\}$ (where $\text{vert}(F)$ is the set of vertices of F) be the poset of the vertex sets of proper faces of P . It can be computed from any vertex-facet incidence matrix of P , since it is the set of all non-empty intersections of the subsets of \mathcal{V} defined by the rows of the incidence matrix (and additionally the empty set). Note that this poset contains the poset $\mathcal{B}(P)$ as a subposet, but may contain additional vertex sets of unbounded faces. It follows that $\varphi' \leq \varphi''$, where φ'' is the size of $\mathcal{P}(P)$; see Remark 13 below for an example which shows that the gap between φ' and φ'' can be large. Moreover, because $\mathcal{B}(P)$ is a polytopal complex, we have the following: if $\text{vert } G \subseteq \text{vert } F$ for an unbounded face G and a bounded face F , there exists a bounded face H such that $\text{vert } H = \text{vert } G$.

Using results of [18], the boundedness of faces can be decided in the following way. The *Möbius number* of a poset element $S \in \mathcal{P}(P)$ is defined as

$$\mu(S) = \begin{cases} 1, & \text{if } S = \emptyset, \\ -\sum_{S' \subsetneq S} \mu(S'), & \text{otherwise,} \end{cases}$$

where the sum ranges over all poset elements $S' \in \mathcal{P}(P)$ strictly less than S . For a face $H \neq P$, we define $\hat{\mu}(H) := \mu(\text{vert } H)$. For $H = P$, we add an artificial top-element $\hat{1}$ to $\mathcal{P}(P)$ and define $\hat{\mu}(P) := \mu(\hat{1})$. Then, the face H

Input: incidence matrix I of P
Output: Hasse diagram \mathcal{D} of $\mathcal{B}(P)$

- 1 initialize \mathcal{D} with N_\emptyset corresponding to the empty face, $\mathcal{M}(\emptyset) = \{(\emptyset, 1)\}$
- 2 initialize queue $\mathcal{Q} \subseteq V(\mathcal{D}) \times 2^V$ by (N_\emptyset, \emptyset)
- 3 **while** $\mathcal{Q} \neq \emptyset$ **do**
- 4 remove first (N_H, H) in \mathcal{Q}
- 5 compute Möbius number $\hat{\mu}(H)$ using $\mathcal{M}(H)$
- 6 **if** $\hat{\mu}(H) \neq 0$ (H is bounded) **then**
- 7 **foreach** vertex v not in H **do**
- 8 $G \leftarrow$ intersection of vertex sets of facets containing $H \cup \{v\}$
- 9 locate/create the node N_G corresponding to G in \mathcal{D}
- 10 **if** N_G was newly created **then**
- 11 add (N_G, G) to \mathcal{Q}
- 12 $\mathcal{M}(G) \leftarrow \mathcal{M}(H)$
- 13 $\mathcal{M}(G) \leftarrow \mathcal{M}(G) \cup \{(H, \hat{\mu}(H))\}$
- 14 add the arc (N_H, N_G) to \mathcal{D}

Algorithm 2: Method using the Möbius function

is bounded if and only if $\hat{\mu}(H) \neq 0$, see [18, Corollary 4.5]. Once all poset elements are known, the entire *Möbius function* of $\mathcal{P}(P)$, that is, the Möbius numbers for all poset elements, can be computed by inverting an appropriate $(\varphi'' \times \varphi'')$ -matrix (the so-called ζ -matrix) in $O((\varphi'')^3)$ time.

We obtain the following algorithm to compute the Hasse diagram of $\mathcal{B}(P)$. Since $\mathcal{P}(P)$ is atomic, its Hasse diagram can be generated by the algorithm in [20] in time $O(n \cdot \beta \cdot \varphi'')$, where β is the number of vertex-facet incidences of the polyhedron P (excluding unbounded information). Then the vertex sets corresponding to unbounded faces are removed by using the Möbius function. In total, we obtain an $O(\max\{n \cdot \beta, (\varphi'')^2\} \cdot \varphi'')$ time algorithm. Since $\mathcal{P}(P)$ is also co-atomic, one can again apply the algorithm of [20] to the dual, which then yields an $O(\max\{m \cdot \beta, (\varphi'')^2\} \cdot \varphi'')$ time algorithm. One can often improve these running times as follows.

Theorem 7. *Given the vertex-facet incidences of P (without information on unbounded edges), the Hasse diagram of $\mathcal{B}(P)$ can be computed in time $O(\max\{n^2 \cdot \varphi', n \cdot \beta\} \cdot n \cdot \varphi')$.*

Proof. Algorithm 2 is a modification of Algorithm 1, in which the Möbius numbers $\hat{\mu}(H)$ are computed on the fly in Step 5. To this end, we use a set $\mathcal{M}(H)$ that stores all elements of $\mathcal{P}(P)$ strictly below H and their corresponding Möbius numbers. Then Step 5 is a straight-forward summation. Throughout this algorithm, the faces G and H of P are encoded as sets of their vertices.

To update $\mathcal{M}(H)$ correctly, we perform a breadth-first search (BFS) over the Hasse diagram of $\mathcal{P}(P)$ from bottom to top, by organizing \mathcal{Q} in the algorithm as a queue. We propagate the necessary information for the computation of Möbius functions in Steps 12 and 13. It follows that once a face H

leaves \mathcal{Q} in Step 4, $\mathcal{M}(H)$ contains all elements of $\mathcal{P}(P)$ strictly below H and their Möbius functions.

For the correctness of the entire algorithm it is essential that $\hat{\mu}(H)$ is computed correctly in Step 5. This is a consequence of two facts: First, $\hat{\mu}(H)$ is computed after all elements of $\mathcal{P}(P)$ below H have been processed; this is different from Algorithm 1. Second, the data structure $\mathcal{M}(H)$ may also contain vertex sets of unbounded faces, but their contribution to $\hat{\mu}(H)$ is zero, by [18, Corollary 4.5]. Therefore, $\hat{\mu}(H)$ evaluates to the Möbius number of H in the poset $\mathcal{P}(P)$.

If H is bounded, only its covering elements G in $\mathcal{P}(P)$ are generated in Step 8. More precisely, G is generated from H by adding one vertex v not contained in H and computing the *closure* with respect to I ; that is, G is the intersection of all vertex sets of facets containing $H \cup \{v\}$; see [20, §2.2]. This is the only situation how new elements can enter the queue. Hence, the while-loop in Step 3 is executed for each of the φ' bounded faces plus at most $n \cdot \varphi'$ faces that are unbounded.

The above discussion implies that $|\mathcal{M}(H)| \leq n \cdot \varphi'$, and hence Step 5 takes $O(n \cdot \varphi')$ time. Note that the size of $\mathcal{M}(H)$ is $O(n^2 \cdot \varphi')$, as we store the vertex set of each face as a bitset. The sets $\mathcal{M}(G)$ can be organized using a face-tree. As observed in KP Lemma 3 can be used in this context as well, where we have to replace α by β and \bar{n} by n . To add information to $\mathcal{M}(G)$ in Step 13, we need to check whether the subset H is present in $\mathcal{M}(G)$, which can then be done in time $O(n \cdot \beta)$. To copy $\mathcal{M}(H)$ in Step 12, we need $O(n^2 \cdot \varphi')$ time.

All other steps are as in Algorithm 1 and their analysis is similar as in the proof of Theorem 4. The only difference is that α can be replaced by β , that is, it is easy to see that by using the vertex-facet incidence matrix of P , we only generate sets in $\mathcal{P}(P)$. In total, we obtain an $O(\max\{n^2 \cdot \varphi', n \cdot \beta\} \cdot n \cdot \varphi')$ time algorithm. \square

Remark 8. Each face-tree data structure needs $O(n^2 \cdot \varphi')$ space, see Remark 5. In total, the algorithm requires $O(n^3 \cdot (\varphi')^2)$ space for maintaining the sets $\mathcal{M}(H)$. If faces are stored as bitsets, the queue \mathcal{Q} needs at most $O(n^2 \cdot \varphi')$ space. The output amounts to an additional space requirement of $O(n \cdot \varphi')$. Thus, we need a total amount of storage of $O(n^3 \cdot (\varphi')^2)$.

3.3. Polyhedra Given in Terms of Inequalities. As mentioned above, the bounded subcomplex $\mathcal{B}(P)$ can be obtained by removing from $\mathcal{L}(\overline{P})$ all *unbounded* faces, that is, faces that contain a vertex of F_∞ .

If the defining inequalities of \overline{P} are given, that is, $\overline{P} = \{x : Ax \leq b\}$, the face lattice of \overline{P} can be computed by an algorithm of Fukuda et al. [12] in time $O(\overline{m} \cdot \ell(A, b) \cdot \varphi)$, where $\ell(A, b)$ is the time to solve a linear program with input size equal to the size of A and b ; its space complexity is $O(\varphi \cdot \log \overline{m} + p(A, b))$, where $p(A, b)$ is the space needed to solve a linear program of the size of A and b . This algorithm outputs the faces as the sets of facets they are contained in. It is also possible to apply the algorithm to the unbounded polyhedron P . Note that the algorithm does not produce the Hasse diagram, but it can be computed using the algorithm in [20].

After the generation of $\mathcal{L}(\overline{P})$, one computes the set of vertices of F_∞ and removes the unbounded faces. Additional work is necessary if the faces

of $\mathcal{B}(P)$ should be given by their vertex sets. This approach via the entire face lattice of \overline{P} is not efficient when the bounded subcomplex $\mathcal{B}(P)$ is much smaller. It is not obvious to the authors whether or not the algorithm from [12] can be modified to compute the bounded faces only.

The above algorithm avoids the explicit computation of the vertex-facet incidences (with unknown complexity) and leads to a polynomial total time algorithm if $\varphi' \approx \varphi$. In practice or if $\varphi' \ll \varphi$, Algorithm 1 might be faster.

3.4. Polyhedra Given in Terms of Vertices and Rays. If the unbounded polyhedron is given by the list of its vertices and rays, one can proceed analogously to the previous section by applying the algorithm of Fukuda et al. [12] to the dual of \overline{P} . It is also easy to adapt their algorithm to work with vertices as input (their “restricted face of polyhedron” problem can also be solved via a linear program in this case). The faces are then given by their vertices and rays. This approach has the same drawbacks as the one in the previous section and can also be applied directly to P .

The affine hull of each bounded k -face of P is spanned by $k + 1$ affinely independent vertices. For each $(k + 1)$ -tuple of the n vertices of P , one can solve one linear program to decide whether there exists a supporting hyperplane containing those $k + 1$ vertices and whose intersection with P is k -dimensional. This immediately yields the following.

Proposition 9. *Let P be given in terms of vertices and rays, and let δ be a fixed constant. Then the set of bounded faces up to dimension δ can be computed in $O(n^\delta \cdot \ell(\mathcal{V}, \text{const}))$ time.*

This algorithm does not produce the Hasse diagram of the δ -skeleton of $\mathcal{B}(P)$ directly, but this can be achieved via the algorithm in [20]. Alternative algorithms would result from computing the facets and the corresponding incidences and then applying the previously mentioned algorithms.

Note that Algorithms 1 and 2 can easily be modified to produce the δ -skeleton as well.

4. SIMPLE POLYHEDRA

In this section, we deal with the special case of simple polyhedra. A pointed d -dimensional polyhedron P is *simple* if each vertex is contained in precisely d facets. Note that \overline{P} may not be simple, even if P is. Nevertheless, using a suitable generic construction one can guarantee that the corresponding polytope is simple, see [15, Prop. 2.2].

For simple polyhedra, Algorithm 1 can be implemented more efficiently. Step 5 can be performed in time $O(d \cdot \alpha)$, and the for-loop is executed at most $O(d \cdot \varphi')$ times, see [20] for more details. Note that the number of vertex-facet incidences is $\alpha = d \cdot \overline{n}$. Thus, we obtain the following.

Proposition 10. *Given a simple polyhedron P and the vertex-facet incidences of \overline{P} , the (Hasse diagram of the) bounded subcomplex of P can be computed in time $O(d \cdot \alpha \cdot \varphi') = O(d^2 \cdot \overline{n} \cdot \varphi')$ time.*

If a simple polyhedron P is given in terms of inequalities the reverse search algorithm by Avis and Fukuda [3] generates the vertices of P in $O(d \cdot m)$ time per vertex, where d is the dimension of P . This is possible by performing

Input: inequality description of a *simple*, pointed, unbounded polyhedron P , far face F_∞

Output: $(f_0, \dots, f_d) = f$ -vector of $\mathcal{B}(P)$

- 1 compute the vertices and the vertex-edge graph Γ of \overline{P}
- 2 find c such that linear program $\max\{c^T x : x \in P\}$ takes its maximum on F_∞ and which is generic on vertices of \overline{P}
- 3 direct the edges of Γ along increasing c
- 4 $d \leftarrow \dim P$
- 5 **for** $k \leftarrow d, d-1, \dots, 0$ **do**
- 6 $h_k \leftarrow$ number of vertices of P with out-degree k
- 7 $\overline{h}_k^\infty \leftarrow$ number of vertices of F_∞ with in-degree k
- 8 $f_k \leftarrow \sum_{i=k}^d \binom{i}{k} (h_i - \overline{h}_i^\infty)$

Algorithm 3: Face numbers of simple polyhedra

a ratio test that decides whether an edge is unbounded or not; see Avis [1] for details. During the vertex enumeration, the vertex-facet incidences of P and \overline{P} can be computed on-the-fly at no extra cost. Using the incidences for \overline{P} , we get the following.

Corollary 11. *Given a simple polyhedron P in inequality form, (the Hasse diagram of) $\mathcal{B}(P)$ can be computed in time $O(\max\{\overline{m}, d \cdot \varphi'\} \cdot d \cdot \overline{n})$.*

Kalai gave an algorithm to compute the vertex-facet incidences of a simple polytope from its vertex-edge graph [21]. This method can be modified to compute the f -vector $(f_0, f_1, \dots, f_{d-1})$ of the bounded subcomplex of a simple d -polyhedron, where f_k is the k -th *face number*, that is, the number of faces of dimension k . Algorithm 3 presents this approach.

Theorem 12. *Algorithm 3 computes the face numbers of a simple polyhedron in $O(d \cdot \overline{m} \cdot \overline{n})$ time.*

Proof. The reverse search algorithm [3] produces the graph Γ of \overline{P} in time $O(d \cdot \overline{m} \cdot \overline{n})$. The running time of Step 1 dominates the remaining steps.

Knowing the vertices of \overline{P} and the far face is equivalent to knowing the vertices and rays of P . Each ray describes a direction in which P is unbounded. Such a ray can be perturbed such that the corresponding linear objective function c takes distinct values on distinct vertices (see Step 2).

The numbers (h_0, h_1, \dots, h_d) computed in Step 6 form the h -vector of \overline{P} , compare [21]. Its relationship with the f -vector, expressed in Step 8, is as follows: Each k -face of \overline{P} has a unique minimal vertex with respect to c . Conversely, each k -set of arcs which leave a fixed vertex v spans a k -face such that v the minimum with respect to c is attained at v . We have to ignore the unbounded k -faces of P , which are precisely the k -faces of \overline{P} whose maximum with respect to c is attained at some vertex of F_∞ . \square

The formula in [15, Prop. 2.4] for the relationship between the f - and h -vectors of an unbounded polyhedron is wrong. The correct version is in Step 8 of Algorithm 3.

5. COMPUTATIONAL RESULTS

The following experiments were performed with the `polymake` system [13], version 2.9.8. The hardware used was an AMD Athlon 64 X2 Dual Core Processor 4200 (4435.84 bogomips) with 4GB main memory running Debian Linux. We tested Algorithm 1 requiring the vertex-facet incidences of \overline{P} as input only. The timings for the convex hull computations required are not given since discussing the various choices is a topic of its own; see [2, 17]. Here we are focusing on the combinatorial aspects. Usually the convex hull computation takes much less time than the computation of the face lattice.

The goals of our computations are threefold.

- The limits of our algorithm can be estimated.
- Several examples for which the computation of $\mathcal{B}(P)$ is interesting are investigated and corresponding results are presented.
- A rough estimation of the sizes up to which a computation of $\mathcal{B}(P)$ is sensible, independent of the approach, can be derived (for our examples).

Our theoretical analysis so far was based on the assumption that sets are stored as bitsets; `polymake` offers a suitable data type (wrapping an implementation of the GMP [14]). However, the representation of sets via balanced trees seems to be superior in a typical scenario. This is the one used in the tests below.

5.1. Dwarfed Cubes. The *dwarfed d -cube* is the polytope

$$\overline{D} := \left\{ x \in [0, 1]^d : \sum x_i \leq \frac{3}{2} \right\}.$$

The polytope \overline{D} has $\overline{m} = 2d + 1$ facets and $\overline{n} = d^2 + 1$ vertices. Moreover, $\alpha = d \cdot \overline{n}$ as the polytope is simple. The interest in these polytopes comes from the fact that the dwarfed cubes provide difficult input to some classes of convex hull algorithms, see Avis et al. [2]. To produce an unbounded polyhedron D , we send the *dwarfing facet* $\sum x_i = 3/2$ to infinity by reversing the construction from Proposition 1. The dwarfing facet contains $d(d - 1)$ vertices; hence the bounded subcomplex has $n = d^2 + 1 - d(d - 1) = d + 1$ vertices and $\beta = n \cdot d = d^2 + d$ incidences between vertices and facets. The bounded subcomplex $\mathcal{B}(D)$ is a star-like graph with $d + 1$ nodes (and d edges); that is, the number φ' of bounded faces equals $2d + 2$, including the empty face.

Table 1 contains the results of the computations. They show that $\mathcal{B}(P)$ could be computed up to a very high dimension (≥ 70) with a moderate computing time; the key reason for this seems to be the small number of bounded faces φ' .

Remark 13. In order to count the faces of the (unbounded) dwarfed cube D we can employ Algorithm 3: To this end consider the linear objective function $\sum x_i$ which is generic on the unbounded dwarfed cube, and which takes its maximum on the dwarfing facet. This linear objective function gives a direction on each bounded edge or ray of D . The origin is the unique node of out-degree d , making up for 2^d non-empty faces of D whose minimum with respect to $\sum x_i$ is 0. The d neighbors of 0, the d unit vectors, are nodes of

TABLE 1. Results for Dwarfed cubes

d	\overline{m}	\overline{n}	α	φ'	time (s)
5	11	26	130	12	0.07
10	21	101	1010	22	0.11
15	31	226	3390	32	0.42
20	41	401	8020	42	1.87
25	51	626	15650	52	6.89
30	61	901	27030	62	28.35
35	71	1226	42910	72	72.93
40	81	1601	64040	82	159.64
45	91	2026	91170	92	324.86
50	101	2501	125050	102	593.73
55	111	3026	166430	112	1039.30
60	121	3601	216060	122	1743.58
65	131	4226	274690	132	2811.10
70	141	4901	343070	142	4457.96
75	151	5626	421950	152	6823.86

out-degree $d-1$, making up for an additional $d \cdot 2^{d-1}$ non-empty faces. Thus $\varphi(D) = 2^d + d \cdot 2^{d-1} + 1$ (including \emptyset and D).

The poset $\mathcal{P}(D)$ contains $2^d - 1$ elements for faces of the first kind above (not containing the face D itself) and d elements corresponding to the d unit vectors. Thus $\varphi'' = 2^d + d$ (including \emptyset). In contrast, φ' is only linear in d .

5.2. Tight Spans of Metric Spaces. The *tight span* (or injective hull) T_M of a finite metric space $M : [d] \times [d] \rightarrow \mathbb{R}$ (see Dress [11] and Isbell [16]) is defined as the bounded complex of the polyhedron

$$P_M = \left\{ x \in \mathbb{R}^d : x_i + x_j \geq M(i, j) \text{ for all } 1 \leq i, j \leq d \right\}.$$

If M is generic enough, as in our examples, all inequalities define facets, hence $\overline{m} = \frac{d(d+1)}{2} + 1$ (including F_∞). It was remarked by Sturmfels and Yu [25] that T_M is dual to the complex of inner faces of the regular subdivision of the *second hypersimplex*

$$\Delta(2, d) := \text{conv} \{ e_i + e_j : 1 \leq i \leq j \leq d \},$$

obtained from interpreting M as a height function. Tight spans are relevant for applications in algorithmic biology, more precisely in phylogenetics, cf. Dress et al. [10, 11].

Two ways to obtain special metric spaces and corresponding examples are described next.

5.2.1. Thrackle Metric. A special triangulation of $\Delta(2, d)$, called the *thrackle triangulation* was introduced by Stanley [24] and thoroughly investigated by De Loera et al. [8]. It turns out that the corresponding metric maximizes the number of faces of the tight span [15] and is equivalent to the tight span of the *maximal circular split system*, see Bandelt and Dress [4, Section 3]. From [15, Theorem 5.5], we deduce that $\overline{n} = 2^{d-1} + d$ and

$$\varphi' = \frac{1}{2} \left((1 - \sqrt{2})^d + (1 + \sqrt{2})^d \right) + 1.$$

TABLE 2. Results for thrackle metrics

d	\overline{m}	\overline{n}	α	φ'	time (s)
3	7	7	24	8	0.01
4	11	12	60	18	0.01
5	16	21	135	42	0.01
6	22	38	288	100	0.03
7	29	71	602	240	0.15
8	37	136	1248	578	0.96
9	46	265	2565	1394	7.38
10	56	522	5210	3364	61.90
11	67	1035	10450	8120	559.08
12	79	2060	20736	19602	5239.04
13	92	4109	40820	47322	54302.46

TABLE 3. Results for random metrics

d	\overline{m}	\overline{n}	α	φ'	time (s)	stddev (s)
5	16	21.00	135.00	42.00	0.03	0.00
6	22	37.99	287.94	99.92	0.06	0.00
7	29	70.65	599.55	237.20	0.19	0.01
8	37	134.95	1247.60	568.96	1.17	0.09
9	46	261.94	2609.46	1365.28	9.52	0.99
10	56	513.55	5495.50	3275.68	86.69	9.69
11	67	1008.46	11588.06	7802.88	841.02	109.44
12	79	1997.28	24627.36	18709.52	9043.11	1351.32

Note that the right hand side is always integral.

Table 2 shows the results. Not surprisingly, it turns out that with increasing dimension the computation time drastically increases along with the number of bounded faces.

5.2.2. Random Metrics. Our random metrics are obtained by taking the distances $M(i, j)$ to be uniformly distributed in the interval $[1, 2]$. Since this is generic (with probability 1), we have $\overline{m} = \frac{d(d+1)}{2} + 1$. The sample size for each dimension where 100 metrics. For \overline{n} , α , and φ' , we state the arithmetic mean, and for the computation time the mean together with the standard deviation.

Table 3 presents the results. Compared to the thrackle metrics (Table 2), random metrics have fewer bounded faces, but their computation times are larger.

5.3. Tropical Polytopes. Let $V = (v_{ik})$ be an $s \times t$ -matrix with real coefficients. We define the polyhedron

$$E_V := \{(u, w) : u_i + w_k \leq v_{ik}\},$$

where $u \in \mathbb{R}^s$ and $w \in \mathbb{R}^t$. Considering (u, w) with sufficiently small coordinates, one can see that E_V is not empty. Moreover, if $(u, w) \in E_V$ then for all $\lambda \in \mathbb{R}$ we have $(u + \lambda \mathbf{1}, w - \lambda \mathbf{1}) = (u, w) + \lambda(\mathbf{1}, -\mathbf{1}) \in E_V$. Hence the one-dimensional subspace $\mathbb{R}(\mathbf{1}, -\mathbf{1})$ is contained in the lineality space of E_V , and

TABLE 4. Results for tropical cyclic polytopes

(s, t)	d	\overline{m}	\overline{n}	α	φ'	time (s)
(3,3)	5	10	12	72	14	0.04
(4,4)	7	17	28	244	64	0.04
(5,5)	9	26	80	840	322	0.40
(6,6)	11	37	264	3144	1684	17.52
(7,7)	13	50	938	12614	8990	1198.35
(8,8)	15	65	3448	52392	48640	139091.23
(3,10)	12	31	68	1003	182	0.22
(3,20)	22	61	233	5903	762	9.17
(3,30)	32	91	498	17703	1742	110.16
(3,40)	42	121	863	39403	3122	814.14
(3,50)	52	151	1328	74003	4902	4418.48
(3,60)	62	181	1893	124503	7082	15595.14
(3,65)	67	196	2213	156653	8322	26858.18
(3,70)	72	211	2558	193903	9662	44651.71

hence we can consider E_V as a polyhedron in the quotient $\mathbb{R}^{s+t}/\mathbb{R}(\mathbb{1}, -\mathbb{1})$. The polyhedron E_V in $\mathbb{R}^{s+t}/\mathbb{R}(\mathbb{1}, -\mathbb{1})$ is pointed, and projecting its bounded subcomplex to \mathbb{R}^t (or, alternatively, to \mathbb{R}^s) yields the *tropical polytope* defined by V ; see Develin and Sturmfels [9]. The bounded subcomplex of E_V is dual to the regular subdivision of the product of simplices $\Delta_{s-1} \times \Delta_{t-1}$, obtained by interpreting the matrix V as a lifting function. Here Δ_r denotes a simplex of dimension r .

We now translate the parameters for tropical polytopes into the parameters that we used in our algorithms. The dimension of the polyhedron E_V (in the quotient) equals $d = s + t - 1$. Its number m of facets is less than or equal to $s \cdot t$. Throughout we have $\overline{m} = m + 1$. The number n of vertices satisfies $n \leq \binom{s+t-2}{s-1}$.

5.3.1. Tropical Cyclic Polytopes. The *tropical cyclic polytope* with parameters (s, t) is given by the matrix $V = (v_{ik})$ with $v_{ik} = i \cdot k$. The corresponding subdivision of $\Delta_{s-1} \times \Delta_{t-1}$ is known as the *staircase triangulation*; see Block and Yu [6]. The polyhedron E_V is simple in this case.

5.3.2. Tropical Permutohedra. Each permutation σ on the t numbers from 0 to $t-1$ can be identified with the vector $(\sigma(0), \sigma(1), \dots, \sigma(t-1))$. This way each permutation contributes one row of a $(t!) \times t$ -matrix V . We call the corresponding tropical polytope a *tropical permutohedron*. The polyhedron E_V is not simple for $t \geq 3$.

TABLE 5. Tropical permutohedra

(s, t)	d	\overline{m}	\overline{n}	α	φ'	time (s)
(6,3)	8	19	24	261	50	0.05
(24,4)	17	97	152	6532	1424	9.07
(120,5)	124	601	1420	276725	76282	143535.58

Table 4 shows the results for tropical cyclic polytopes and Table 5 for tropical permutohedra. Both cases show a steep increase in computation time with increasing dimension. The remarkable fact is that examples of these sizes can be handled at all.

6. CONCLUDING REMARKS AND OPEN QUESTIONS

We presented combinatorial algorithms to compute (the Hasse diagram of) the bounded faces of an unbounded pointed polyhedron. Algorithm 1, which takes the vertex-facet incidences of \overline{P} as input, was also shown to work in practice via extensive computations. It seems that the examples which we presented in the last section show the limits of what one can currently compute in acceptable time, unless some (possibly fundamentally) different idea comes up. For instance, a central open question is the following.

Question. *Is there a polynomial total time algorithm to compute $\mathcal{B}(P)$ for general P , given the inequalities?*

For instance, this would follow if it were possible to modify the algorithm of Fukuda et al. [12] to compute the bounded faces only.

In order to get an idea about the size of a bounded subcomplex it would be interesting if it were possible to compute statistical information without generating all faces.

Question. *Is there a polynomial time algorithm to compute the f -vector of $\mathcal{L}(P)$ or $\mathcal{B}(P)$?*

See the discussion in Section 4 for simple polyhedra.

REFERENCES

1. David Avis, *A revised implementation of the reverse search vertex enumeration algorithm*, Polytopes – Combinatorics and Computation (Gil Kalai and Günter M. Ziegler, eds.), DMV Seminar, vol. 29, Birkhäuser, Basel, 2000, pp. 177–198.
2. David Avis, David Bremner, and Raimund Seidel, *How good are convex hull algorithms?*, Comput. Geom. **7** (1997), no. 5-6, 265–301, 11th ACM Symposium on Computational Geometry (Vancouver, BC, 1995). MR1447243 (98c:52017)
3. David Avis and Komei Fukuda, *A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra*, Discrete Comput. Geom. **8** (1992), no. 3, 295–313, ACM Symposium on Computational Geometry (North Conway, NH, 1991). MR1174359 (93h:68137)
4. Hans-Jürgen Bandelt and Andreas W. M. Dress, *A canonical decomposition theory for metrics on a finite set*, Adv. Math. **92** (1992), no. 1, 47–105. MR1153934 (93h:54022)
5. Roswitha Blind and Peter Mani-Levitska, *Puzzles and polytope isomorphisms*, Aequationes Math. **34** (1987), no. 2-3, 287–297. MR921106 (89b:52008)
6. Florian Block and Josephine Yu, *Tropical convexity via cellular resolutions*, J. Algebraic Combin. **24** (2006), no. 1, 103–114. MR2245783
7. Endre Boros, Khaled Elbassioni, Vladimir Gurvich, and Hans Raj Tiwary, *The negative cycles polyhedron and hardness of checking some polyhedral properties*, Annals of OR (2010), to appear.
8. Jesús A. De Loera, Bernd Sturmfels, and Rekha R. Thomas, *Gröbner bases and triangulations of the second hypersimplex*, Combinatorica **15** (1995), no. 3, 409–424. MR1357285 (97b:13035)
9. Mike Develin and Bernd Sturmfels, *Tropical convexity*, Doc. Math. **9** (2004), 1–27 (electronic), erratum ibid., pages 205–206. MR2054977 (2005i:52010)

10. Andreas Dress, Vincent Moulton, and Werner Terhalle, *T-theory - an overview*, Europ. J. Combinatorics **17** (1995), 161–175.
11. Andreas W. M. Dress, *Trees, tight extensions of metric spaces, and the cohomological dimension of certain groups: a note on combinatorial properties of metric spaces*, Adv. in Math. **53** (1984), no. 3, 321–402. MR753872 (86j:05053)
12. Komei Fukuda, Thomas M. Liebling, and François Margot, *Analysis of backtrack algorithms for listing all vertices and all faces of a convex polyhedron*, Comput. Geom. **8** (1997), no. 1, 1–12. MR1452921 (98b:68187)
13. Evgenij Gawrilow and Michael Joswig, *polymake: a framework for analyzing convex polytopes*, Polytopes—combinatorics and computation (Oberwolfach, 1997), DMV Sem., vol. 29, Birkhäuser, Basel, 2000, pp. 43–73. MR1785292 (2001f:52033)
14. Torbjörn Granlund et al., *The GNU multiple precision arithmetic library*, gmplib.org.
15. Sven Herrmann and Michael Joswig, *Bounds on the f-vectors of tight spans*, Contrib. Discrete Math. **2** (2007), no. 2, 161–184 (electronic). MR2358269
16. John R. Isbell, *Six theorems about injective metric spaces*, Comment. Math. Helv. **39** (1964), 65–76. MR0182949 (32 #431)
17. Michael Joswig, *Beneath-and-beyond revisited*, Algebra, geometry, and software systems, Springer, Berlin, 2003, pp. 1–21. MR2011751 (2004k:68169)
18. Michael Joswig, Volker Kaibel, Marc E. Pfetsch, and Günter M. Ziegler, *Vertex-facet incidences of unbounded polyhedra*, Adv. Geom. **1** (2001), no. 1, 23–36. MR1823950 (2002b:52015)
19. Michael Joswig and Thorsten Theobald, *Algorithmische Geometrie – Polyedrische und algebraische Methoden*, Vieweg-Verlag, 2008.
20. Volker Kaibel and Marc E. Pfetsch, *Computing the face lattice of a polytope from its vertex-facet incidences*, Comput. Geom. **23** (2002), no. 3, 281–290. MR1927137 (2003h:52019)
21. Gil Kalai, *A simple way to tell a simple polytope from its graph*, J. Combin. Theory Ser. A **49** (1988), no. 2, 381–383. MR964396 (89m:52006)
22. Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled Elbassioni, and Vladimir Gurvich, *Generating all vertices of a polyhedron is hard*, Discrete Comput Geom **39** (2008), 174–190.
23. Raimund Seidel, *Convex hull computations*, Handbook of Discrete and Computational Geometry (Jacob Goodman and Joseph O’Rourke, eds.), CRC Press, Boca Raton, 1997, pp. 361–375.
24. Richard P. Stanley, *Eulerian partitions of the unit hypercube*, p. 49, D. Reidel, 1977.
25. Bernd Sturmfels and Josephine Yu, *Classification of six-point metrics*, Electron. J. Combin. **11** (2004), no. 1, Research Paper 44, 16 pp. (electronic). MR2097310 (2005m:51016)
26. Günter M. Ziegler, *Lectures on polytopes*, Springer-Verlag, New York, 1995, Revised edition 1998.

SVEN HERRMANN, SCHOOL OF COMPUTING SCIENCES, UNIVERSITY OF EAST ANGLIA, NORWICH, NR4 7TJ, UK

E-mail address: s.herrmann@uea.ac.uk

MICHAEL JOSWIG, FACHBEREICH MATHEMATIK, TU DARMSTADT, 64289 DARMSTADT, GERMANY

E-mail address: joswig@mathematik.tu-darmstadt.de

MARC E. PFETSCH, INSTITUTE FOR MATHEMATICAL OPTIMIZATION, TU BRAUNSCHWEIG, 38106 BRAUNSCHWEIG, GERMANY

E-mail address: m.pfetsch@tu-bs.de